COBB INSIDE DOS®™

Tips & techniques for MS-DOS & PC-DOS Versions 5 and 6

July 1993
Vol. 4 No. 7

# Redefining commands with DOSKEY

VERSION 5.0 & 6.0

By Van Wolverton

*This month, Van Wolverton discusses ways to make some powerful new commands in DOS 6.0 safer to use. His technique—redefining dangerous commands as DOSKEY macros—also works for redefining other commands such as FORMAT and MEM, which are available in DOS 5.0.*

DOS is the most important program we use. Without DOS, none of our other programs—not even Windows—can run. The better we know how to use DOS, the more efficiently we can use our computers and the more we can tailor them to our specific needs. That's why we spend time learning to use DOS.

But it's also important to learn something about the shortcomings and pitfalls of DOS. Have you ever typed *del *.bat* when you meant to type *del *.bak*? Have you ever lost a file by copying another on top of it or by redirecting the output of a command to a file that already exists? Have you ever typed *del *.** when you were in the wrong directory?

Some of these mistakes are clearly our own, such as typing *del *.bat* for *del *.bak*, but many of the others are more properly the responsibility of the designers of DOS, who chose to make the DELETE command a silent killer (offering no confirming prompt unless you're deleting all the files in a directory). The designers also chose to create the even more insidious trap of letting the COPY and XCOPY commands overwrite an existing file with no warning (unless you're using the DOS Shell and choose *Confirm on Replace* from the Confirmation option).

Now, with Version 6.0 of DOS, we've gained more sharp instruments to use with care. The MOVE command, like COPY and XCOPY, will destroy any target files with the same name without a hint of what's about to happen. The DELTREE command, while providing a long-awaited capability, also gives us the power to wipe out all the files on a hard disk with one command.

## We can protect ourselves

The picture isn't all bleak, of course. DOS does warn us of some potentially disastrous steps, and we can exercise special caution when using the more dangerous commands. In addition, DOS gives us some tools we can use to reduce the risk factor.

For example, in earlier versions of DOS, it was all too easy to format your hard disk inadvertently, losing all the files it contained. Before recent versions added confirming prompts and the UNFORMAT command, many users changed the working of the FORMAT command by renaming FORMAT.COM (XFORMAT.COM or a similar name) and creating a batch file named FORMAT.BAT that offered a safer operation.

But that technique was good only for external DOS commands because we were renaming the command file. There was no simple way to alter internal commands such as DIR and COPY. Then Version 5.0 gave us DOSKEY.

The most frequently used feature of DOSKEY is the ability to reuse previously typed commands. But DOSKEY also lets you create macros. If you create a DOSKEY macro with the same name as an internal command, DOS carries out the macro instead of the internal command whenever you type the name of the macro. (As we showed in "Running a COM, EXE, or BAT File Having the Same Name as a Macro," in the April 1993 issue, you can tell DOS to bypass the macro and carry out the command by typing a space before the command.)

Using DOSKEY macros, then, you can redefine any internal command to suit your particular needs. With

**IN THIS ISSUE**

the additional risk factors introduced in Version 6.0, perhaps the time has come to blunt some of those sharp edges that sometimes make DOS more thrilling to use than we'd like.

## Redefining an internal command

It's easy to redefine an internal command. Type the following DOSKEY command to change the VER command to the VOL command:

```
C:\>doskey ver=vol
```

Now, when you type *ver*, DOS responds as if you'd typed *vol*:

```
C:\>ver
Volume in drive C is RUBICON ONE
Volume Serial Number is 1A83-9EF8
C:\>
```

As shown, DOS echoes the actual command that's carried out by the macro—in this case, *vol*.

You can use this technique to tailor the default behavior of a command. For example, if you use either the /C or /P parameter of the MEM command to see the details of how your computer is using memory, you'll also have to use the /P parameter to pause the display. The /P parameter has no effect if the output is less than a screenful, so why not redefine MEM to include /P no matter what other parameters are specified? The macro definition is simple:

```
doskey mem=mem /p $*
```

The $* parameter represents everything typed after the macro name. Now whenever you type the MEM command, if the output is more than one screenful, the system will pause after each screenful and wait for you to press a key before displaying the next screenful. If the output is a screenful or less, there's no change.

If you want the command to behave this way each time you use your system, put the DOSKEY command in your AUTOEXEC.BAT.

## A macro must be typed

Like some other parts of DOS, DOSKEY has its idiosyncrasies. The most significant of these is that a DOSKEY macro can be typed only at the command line; you can't use a macro in a batch file or another macro definition. If you create a macro whose name isn't the same as a DOS command and include that macro in a batch file or macro definition, DOS responds with the message *Bad command or file name*. If you create a macro with the same name as an internal command and include that macro in a batch file or macro definition, DOS carries out the *unmodified* command, not the macro you defined.

INSIDE DOS™

*Inside DOS* (ISSN 1049-5320) is published monthly by The Cobb Group.

**Prices**

| | | |
|---|---|---|
| Domestic: | $49/yr. ($6.00 each) |
| Outside US: | $69/yr. ($8.50 each) |

**Phone**

| | |
|---|---|
| Toll free: | (800) 223-8720 |
| Local: | (502) 491-1900 |
| Customer Relations Fax: | (502) 491-8050 |
| Editorial Department Fax: | (502) 491-4200 |

**Address**

You may address tips, special requests, and other correspondence to

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

**Postmaster**

Second class postage is paid in Louisville, KY. Send address changes to

*Inside DOS*
P.O. Box 35160
Louisville, KY 40232

**Back Issues**

To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $6.00 each, $8.50 outside the US. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you. Please identify the issue you want by the month and year it was published. Customer Relations can also provide you with an issue-by-issue listing of all the articles that have appeared in *Inside DOS*.

To demonstrate this idiosyncrasy, type the following instruction to redefine the CLS command as the VER command:

```
C:\>doskey cls=ver
```

Now what do you have? A few moments ago, you redefined the VER command as VOL. Now you've redefined CLS as VER. If you type *ver*, DOS will display the volume label as if you had typed *vol*. What will DOS do if you type *cls*? Try it:

```
C:\>cls
C:\>ver

MS-DOS Version 6.00
C:\>
```

DOS carries out the real VER command, not its redefinition as the VOL command, because you didn't type *ver*—it was part of a macro definition. DOS carries out a DOSKEY macro only when you type it at the command prompt.

To restore the original meaning of a command you've redefined, you simply cancel the macro by typing *doskey* followed by the command name and an equal to sign. Type the following two DOSKEY commands to restore the original meanings of the VER and CLS commands:

```
C:\>doskey ver=
C:\>doskey cls=
```

These commands have the same effect as typing *doskey ver=ver* and *doskey cls=cls*.

## Lowering the risk of DELTREE

You can take advantage of this quirk in DOSKEY to redefine a command and change potentially dangerous behavior. DELTREE is one of the more dangerous DOS 6.0 commands. What can you do to reduce the possibility of losing valuable files?

The brute force approach would be to eliminate DELTREE. There are a couple of ways to do so. The simplest approach is just to redefine DELTREE as CLS (*doskey deltree=cls*). When you type the DELTREE command, the only effect is to clear the screen.

A slightly more informative—but equally misleading—way to eliminate the DELTREE command is to redefine DELTREE so that it displays *Bad command or file name*. This technique requires creating a short batch file named DTREE.BAT and redefining DELTREE to carry out the batch file. DTREE.BAT contains only three commands:

```
@echo off
cls
echo Bad command or file name
```

Now, if you redefine DELTREE to carry out DTREE.BAT (*doskey deltree=dtree*), typing a DELTREE command causes the screen to clear and the message *Bad command or file name* to appear at the top, just as if you'd typed something other than a command. (The screen clears to eliminate the clutter caused by DOS echoing the ECHO command.)

In either case, this redefinition would apply only when you typed *DELTREE* at the command prompt. If DELTREE appeared in either a batch file or a macro definition, DOS would carry out the unmodified DELTREE command.

## A more elegant solution

Rather than defeat DELTREE, suppose you want to keep it around but make it safer to use. What if DELTREE could do the following:

- Display the complete list of directories and files that it would delete
- Prompt for confirmation that deleting these directories and files is, indeed, the desired action
- If the response is affirmative, carry out the DELTREE command (which issues yet another confirming prompt)

You can accomplish these objectives by redefining the DELTREE command to run a batch file—the version of the DTREE.BAT shown in Figure A—that carries out these steps. When you run DTREE.BAT and answer yes to its prompt, the batch file carries out a DELTREE command. DOS then carries out the actual DELTREE command, not the redefined version, because the command is in a batch file, not typed at the command prompt.

**Figure A**

```
@echo off
cls
echo You're about to delete the following directories
echo and files:
pause
dir %1 /s /p
echo.
echo If you REALLY want to delete these directories
echo and files, press Y; to cancel the command,
echo press N. If you don't respond within 10 seconds,
choice the DELTREE command will be canceled. /t:n,10
if errorlevel 2 goto :CANCEL
if errorlevel 1 deltree %1
goto :END
:CANCEL
echo DELTREE COMMAND CANCELED
:END
```

*DTREE.BAT is a safer replacement for DELTREE.*

Because the batch file must know what parameter you specified with DELTREE, the DOSKEY command that redefines the DELTREE command includes the $1 parameter, which represents the first parameter typed with the command, just as %1 represents the first parameter typed in a batch file:

```
doskey deltree=dtree $1
```

## How DTREE.BAT works

The first two ECHO commands display a message that says you're about to see the files DELTREE will delete. The following PAUSE command waits for a keypress.

After you press a key, the DIR command displays the directories and files that DELTREE will delete by using %1 to identify the pathname specified with the DELTREE command and the /S parameter to include all subdirectories.

The next four ECHO commands and the CHOICE command (new to DOS 6.0) display a blank line and a message that asks whether you want to delete the directories and files displayed. The CHOICE command displays the last line of the message so that the [Y,N] prompt appears on the same line. If you prefer, you can use an ECHO command to display the last line of the prompt and put the CHOICE command on a line by itself, in which case the [Y,N] prompt would appear on the line following the last line of the prompt. The /T parameter specifies that the default response N is your choice if you don't type a response within ten seconds. You can omit this parameter, in which case the CHOICE command will wait until you type a response.

The first IF command checks to see whether you typed *N* in response to the CHOICE command (or selected it by default). If you did, command processing skips to the ECHO command following the label CANCEL, which displays the message *DELTREE COMMAND CANCELED* and returns to the command prompt.

The next IF command checks to see whether you typed *Y* in response to the CHOICE command. If so, the batch file carries out the original DELTREE command with one parameter—%1—the parameter you typed with the DELTREE command (because the DOSKEY command specified *deltree=dtree $1*). After the batch file carries out the DELTREE command, command processing jumps to the label END, which returns to the DOS command prompt.

Now when you type a DELTREE command, the screen clears and you see the opening message:

```
You're about to delete the following directories
and files:
Press any key to continue...
```

Press a key, and DOS displays the complete list of directories and files in the directory you specified when you typed the DELTREE command. The display pauses after each screenful to give you a chance to read it.

After the last screenful of directory entries appears, you see the confirming prompt:

```
If you REALLY want to delete these directories
and files, press Y; to cancel the command,
press N. If you don't respond within 10 seconds,
the DELTREE command will be canceled. [Y,N]
```

If you press Y, the real DELTREE command executes and you see its confirming prompt. For example, if you'd typed *deltree fred*, the prompt would read

```
Delete directory "fred" and all its subdirectories? [yn]
```

Press Y and then [Enter] to run the DELTREE command.

Your redefined version of DELTREE shows you what it will delete and gives you two chances to change your mind. If you take longer than ten seconds to decide whether you want to delete the files, DOS cancels the command. That's not an unreasonable amount of caution, given the potential consequences.

Remember, if you want this redefinition of the DELTREE command to take effect whenever you use your system, put the DOSKEY command in your AUTOEXEC.BAT file. And chalk up another way that you have made DOS more closely meet your needs.

*Contributing editor Van Wolverton is the author of the best-selling books* Running MS-DOS *and* Supercharging MS-DOS. *Van, who has worked for IBM and Intel, lives in Alberton, Montana.*

## Misplaced carriage return

In the June 1993 issue, we showed you a sample CONFIG.SYS in Figure B of the article "Choosing a Configuration at Startup Is Easier with DOS 6." The first two lines of that file read

```
dos = high
umb device = c:\dos\himem.sys
```

These lines should have read

```
dos = high, umb
device = c:\dos\himem.sys
```

Our carriage return was in the wrong place. We apologize for any inconvenience this error may have caused.

VERSION
5.0 & 6.0

# Using DISKCOPY to make exact copies of your diskettes

Many software installation guides instruct you to copy the installation diskettes and then use the copies instead of the originals to install the program. Working from copies of the diskettes is an excellent idea. First, you'll protect your original copies from any damage that could occur as you install the software. Later, if disaster should strike your hard disk, you'll have two sets of diskettes ready to reinstall your program. For added insurance, you can keep the two sets in different locations. If your office ceiling leaks or you leave a magnetic paper clip holder near one set, you'll have the other set to fall back on. Once you've made the copy, you'll even be protected in case your diskette drive damages one of the diskettes as you install the program.

To simplify the process of copying diskettes, DOS includes the DISKCOPY command. As we'll show in this article, using the DISKCOPY command is usually fairly simple—if you know what to expect. Let's get started by looking at how DISKCOPY differs from other copy commands.

## DISKCOPY's advantages and limitations

DISKCOPY has a well-defined job: making exact copies of diskettes. Because it makes exact, sector-by-sector copies of diskettes, DISKCOPY can perform some functions that the COPY and XCOPY commands can't. Most important, DISKCOPY lets you

- copy all files from a diskette, even hidden and system files
- copy the volume label from a diskette
- make an exact, bootable copy of a boot diskette

However, because DISKCOPY copies data sector by sector, it can only copy to a diskette of the same size. For example, you can't copy a 3.5", 1.44-Mb diskette to a 5.25", 1.2-Mb diskette. You're also limited by the capacity of your diskette drives. If you have high-density drives, you can use DISKCOPY to copy either high-density or double-density diskettes. For example, if you have a high-density, 3.5" diskette drive, you can copy either 1.44-Mb or 720-Kb diskettes. But if you have a double-density drive, you can't copy a high-density diskette. For example, if you have a double-density, 5.25" diskette drive, you can copy 360-Kb diskettes but not 1.2-Mb diskettes. (For more information on diskette capacities, see the article "Choosing the Right Format Option for the Right Diskette," which appeared in the October 1992 issue of *Inside DOS*.) In addition, you can't use DISKCOPY when you want to copy to or from a hard disk.

## Getting ready

When you want to copy a diskette or set of diskettes, you'll need to have ready a diskette (or set of diskettes) of the same size and capacity as the diskette you want to copy. You can use formatted or unformatted diskettes; if you're using an unformatted diskette, DISKCOPY will format the target diskette to the same capacity as the source diskette.

However, DISKCOPY won't reformat an already formatted target diskette to a different capacity. For example, if you're copying a 360-Kb diskette and you insert a target diskette formatted as 1.2 Mb, DISKCOPY will present the message *Drive types or diskette types not compatible* and then quit.

As a precaution, you might want to write-protect your source diskette. Write-protecting the source diskette can save your data if you accidentally reverse the source and target drive letters when you run DISKCOPY or if you inadvertently insert the source diskette when DISKCOPY prompts you for the target diskette. However, write-protecting the source diskette may be unnecessary if you're making backups of installation diskettes, since manufacturers often ship their software on write-protected diskettes. If you choose not to write-protect the source diskette, label it if it isn't already clearly marked as the source.

## Using the DISKCOPY command

Once you've made some preparations, the DISKCOPY command is fairly simple to use. You just type *diskcopy*, the drive that contains the source diskette, and the drive that contains the target diskette. For example, if your A: and B: drives are the same size, you can copy a diskette from one drive to the other. If you place the diskette you want to copy in the A: drive and place a blank diskette in the B: drive, you'd issue the DISKCOPY command

```
C:\>diskcopy a: b:
```

Before you press [Enter], take a moment to be sure you placed the source diskette in the A: drive and the blank diskette in the B: drive. When you press [Enter], the A: drive light will flash on as DISKCOPY copies information from the source diskette into RAM (random access memory). As it copies the data, DISKCOPY will tell you how many tracks and sectors it's copying from the source diskette. For example, if you're copying a 1.44-Mb, 3.5" diskette, DISKCOPY will display the message

```
Copying 80 tracks
18 sectors per track, 2 side(s)
```

When your computer's available RAM fills with data, DISKCOPY will transfer the data to the target diskette in the B: drive. Because RAM can't hold all the information from the diskette at once, DISKCOPY will then copy another chunk of data from the A: drive into RAM and then onto the diskette in the B: drive. The number of reading and writing exchanges will depend on the amount of data on the diskette and the amount of RAM you have free. When DISKCOPY has copied all the sectors from the source diskette to the target diskette, it will present the message

```
Copy another diskette (Y/N)?
```

If you want to copy another diskette, place the source diskette in the A: drive, place a blank target diskette in the B: drive, and press Y. Or, if you wanted to copy only one diskette, press N. DISKCOPY will return you to the DOS prompt.

## Copying a diskette with a single drive

For those of us who have two identical diskette drives, the previous example shows a convenient way to use the DISKCOPY command. However, most people either have only one diskette drive or have A: and B: drives of different sizes. In either case, you'll need to copy the diskette using a single drive. As you might expect, you can do so by specifying the same drive letter as both the source drive and the target drive.

Let's take a closer look at how copying a diskette in a single drive works. Suppose you want to copy a high-density, 5.25" diskette in your A: drive to a blank, high-density, 5.25" diskette. Let's also suppose you've covered the write-protect notch on the source diskette so that you don't risk losing the data you're trying to copy. You can begin copying this diskette by entering the command

```
C:\>diskcopy a: a:
```

The DISKCOPY command will prompt you for the source diskette, then pause:

```
Insert SOURCE diskette in drive A:

Press any key to continue. . .
```

After you've inserted the source diskette into the drive, press a key. The DISKCOPY command will then tell you how many tracks and sectors it will copy from the source diskette:

```
Copying 80 tracks
15 sectors per track, 2 side(s)
```

The A: drive light will flash on as DISKCOPY copies sectors of data from the diskette into RAM. When the available RAM fills, DISKCOPY will prompt you for the target diskette:

```
Insert TARGET diskette in drive A:

Press any key to continue. . .
```

At this point, you'll need to remove the source diskette from the A: drive and insert the unformatted target diskette. When you press a key, DISKCOPY will check to see if the diskette is formatted. Since the diskette isn't formatted, DISKCOPY will format the diskette as it displays the message

```
Formatting while copying
```

Once DISKCOPY formats the diskette, the command will write the sectors it saved in RAM to the target diskette. Then, DISKCOPY will prompt you to enter the source diskette again. Now you remove the target diskette, insert the source diskette in the A: drive, and press a key. Again, DISKCOPY will copy as many sectors from the diskette as possible, then display the *Insert TARGET diskette in drive A:* message. Now, let's suppose you accidentally leave the source diskette in the A: drive. Because you've write-protected the source diskette, DISKCOPY will display the following messages when you press a key:

```
Write protect error

Press CTRL+C to abort,
or correct this problem and press any other key to continue
```

In this case, it's easy to correct the problem and continue. Simply remove the source diskette from the A: drive, replace it with the target diskette, and press a key. DISKCOPY will continue copying the diskette.

You'll need to trade diskettes at least one more time to complete the process—perhaps more if your PC doesn't have much RAM available. Finally, DISKCOPY will tell you the serial number it assigned to the target diskette and ask if you want to copy another diskette:

```
Volume serial number is 16D7-045F

Copy another diskette (Y/N)?
```

Now that you've copied your diskette, you can press N, and DISKCOPY will return you to the DOS prompt.

## Note

DISKCOPY allows you to add the /V switch if you want the command to verify that the data is correctly written to the source diskette. Adding this switch usually isn't necessary. However, if you decide to add the /V switch, you can expect the DISKCOPY process to take at least 20 percent longer than it would without the switch.

## Conclusion

DISKCOPY provides a relatively easy way to copy diskettes. Unlike the DOS COPY and XCOPY commands, DISKCOPY will automatically copy hidden and system files. We've shown you the basics of using DISKCOPY to copy between two diskette drives and with a single diskette drive.

SCREEN DISPLAY TIP

VERSION 5.0 & 6.0

# Changing the screen display with the MODE command

The MODE command serves many functions, such as configuring a printer, displaying the status of various devices, and setting the delay and repeat rate of the keyboard. Yet one of its simplest and most useful functions is often overlooked: The MODE command can change the number of lines and the number of columns your screen displays. How is this useful? Well, suppose you're reading through a long text file containing references to other parts of the file. By increasing the number of lines displayed on the screen, you can reduce the number of screens you have to go through to find the referenced text. In some cases, you might actually find a reference and its corresponding text in the same screen. On the other hand, you can reduce eyestrain by decreasing the number of lines and columns your screen displays.

Whether you prefer your display tall or short, wide or narrow, you'll probably run into instances when a change in the line display might be beneficial. When you do, you'll want to invoke the MODE command.

## Before you change the screen display . . .

Before you can change the display, you must install ANSI.SYS, DOS' extended screen and keyboard control driver. You can install ANSI.SYS by using the command

```
C:\>device=c:\dos\ansi.sys
```

(If ANSI.SYS is in a directory other than the C:\DOS directory, be sure to substitute the correct path.) To install ANSI.SYS every time you boot your computer, just add this line to your CONFIG.SYS file.

## Changing the number of columns your screen displays

Once you've installed the ANSI.SYS device driver, you can use the MODE command to change the number of columns on your screen. To do so, you simply type

```
C:\>mode m
```

where *m* can equal 40 for a 40-column display or 80 for an 80-column display (the default).

## Changing the number of lines your screen displays

Installing the ANSI.SYS device driver also lets you use the MODE command to change the number of lines on your screen. However, you need to use a different form of the MODE command. To change the number of lines on your screen, you type

```
C:\>mode con: lines=n
```

where *n* depends on the type of display you have. If your display is a VGA, *n* can equal 25 for a 25-line display (the default) or 50 for a 50-line display. If your display is an EGA, you can use only a value of 43 for *n*.

## Changing the number of columns and lines at the same time

When you want to change the number of columns *and* the number of lines on your screen, you can use one MODE command to take care of both changes together. The command

```
C:\>mode con: cols=m lines=n
```

allows you to specify the number of columns you want (cols=40 or cols=80) and the number of lines you want (lines=25, lines=43, or lines=50, depending on your display type).

Now, when your work, your eyesight, or your preference calls for a change in the screen display, you can issue the form of the MODE command that suits your needs.

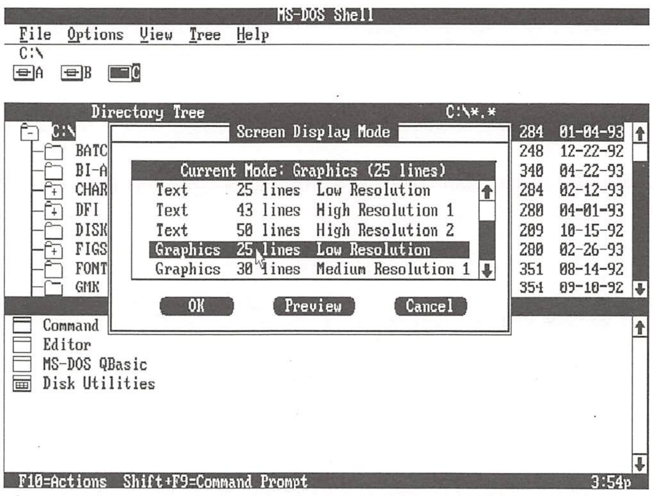
# Customizing the DOS Shell screen

If you're a DOS Shell advocate, you know that its graphical user interface is much easier to understand than DOS' default command-line interface. However, if your Shell screen looks like the one in Figure A, you might not be aware of all the options Shell provides for tailoring the screen to meet your needs. In this article, we'll tell you how to customize the Shell screen to help you design the working environment you're most comfortable with.

## Choosing a screen display mode

DOS Shell offers two types of screen displays: text and graphics. Each type supports several options for the number of lines on the screen. Figure A shows Shell's default screen—a text display of 25 lines. The type and line-display options that are available depend on the type of monitor you use. For example, if your monitor doesn't support graphics, a text display is the only type available. However, you might still have several line-display options.

**Figure C**

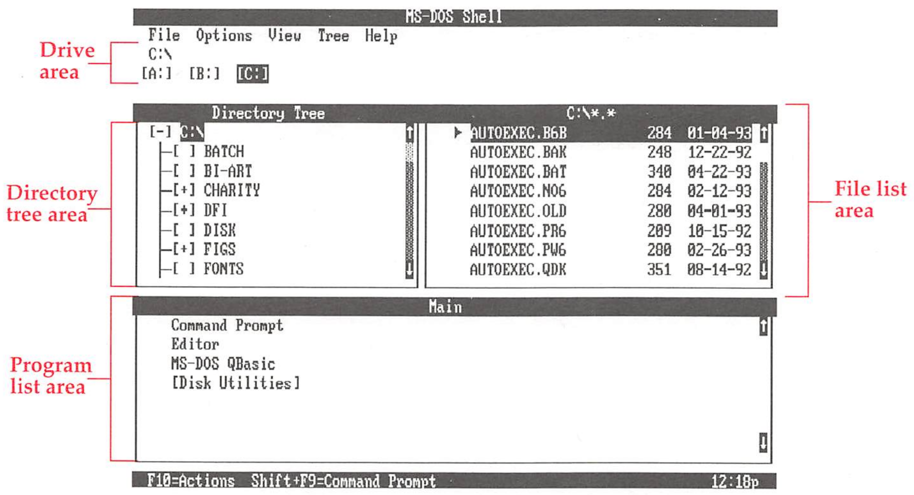


```
                              MS-DOS Shell
 File  Options  View  Tree  Help
 C:\
 A   B   C
     Directory Tree                          C:\*.*
 C:\            Screen Display Mode               284  01-04-93
  BATC                                            248  12-22-92
  BI-A     Current Mode: Graphics (25 lines)      340  04-22-93
  CHAR   Text      25 lines  Low Resolution       284  02-12-93
  DFI    Text      43 lines  High Resolution 1    280  04-01-93
  DISK   Text      50 lines  High Resolution 2    209  10-15-92
  FIGS   Graphics  25 lines  Low Resolution       280  02-26-93
  FONT   Graphics  30 lines  Medium Resolution 1  351  08-14-92
  GMK                                             354  09-10-92
            OK        Preview       Cancel
 Command
 Editor
 MS-DOS QBasic
 Disk Utilities

 F10=Actions  Shift+F9=Command Prompt                       3:54p
```

*You can preview any of the options in the Screen Display Mode dialog box before making your choice.*

**Figure A**

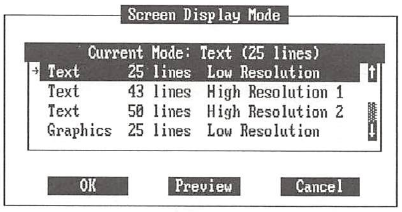


```
                              MS-DOS Shell
 File  Options  View  Tree  Help
 C:\
 [A:]  [B:]  [C:]
        Directory Tree                         C:\*.*
 [-] C:\                         AUTOEXEC.B6B      284  01-04-93
  ├[ ] BATCH                     AUTOEXEC.BAK      248  12-22-92
  ├[ ] BI-ART                    AUTOEXEC.BAT      340  04-22-93
  ├[+] CHARITY                   AUTOEXEC.N06      284  02-12-93
  ├[+] DFI                       AUTOEXEC.OLD      280  04-01-93
  ├[ ] DISK                      AUTOEXEC.PR6      209  10-15-92
  ├[+] FIGS                      AUTOEXEC.PW6      280  02-26-93
  └[ ] FONTS                     AUTOEXEC.QDK      351  08-14-92
                                 Main
   Command Prompt
   Editor
   MS-DOS QBasic
   [Disk Utilities]

 F10=Actions  Shift+F9=Command Prompt                       12:18p
```

*You can customize DOS Shell's default screen to suit your needs and preferences.*

**Figure B**



```
              Screen Display Mode
       Current Mode: Text (25 lines)
   Text      25 lines  Low Resolution
   Text      43 lines  High Resolution 1
   Text      50 lines  High Resolution 2
   Graphics  25 lines  Low Resolution

      OK         Preview        Cancel
```

*The Screen Display Mode dialog box lists the display modes your monitor supports.*

To view your screen display options, you issue the Display... command from the Options menu by pressing [Alt]O,D. Doing so opens the Screen Display Mode dialog box, shown in Figure B. The heading of the list box in this dialog box indicates the currently selected display mode. You can press ↑ or ↓ or click the scroll bar to view the other display options.

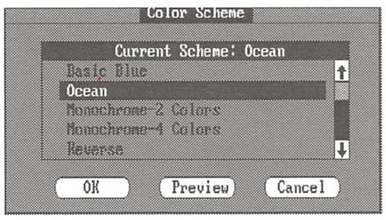
Before you decide on a display mode, you can see how your screen will look in that mode. You simply highlight the mode by pressing the arrow keys; then, you click the Preview button. Figure C shows a preview of a 25-line graphics display mode. Once you're satisfied with the display mode you've selected, you click OK to return to the DOS Shell screen.

If you compare the graphics display in Figure C with the text display in Figure A, you can see that the icons in graphics mode clearly represent the items listed. Unless you're familiar with what the icons in text mode represent, you might have trouble deciphering their meanings. However, graphics modes require more memory than text modes, so you'll want to go with a text mode if you experience memory problems.

## Selecting a color scheme

Besides letting you choose between text and graphics modes, DOS Shell lets you select one of several built-in color schemes. To view your color scheme options, issue the Colors... command from the Options menu. This command opens the Color Scheme dialog box, shown in Figure D.

**Figure D**



*The Ocean color scheme is Shell's default.*

As you can see in Figure D, the Color Scheme dialog box is set up just like the Screen Display Mode dialog box. The heading on the list box tells you that Ocean is the currently selected color scheme. The list box also offers Basic Blue, Monochrome-2 Colors, Monochrome-4 Colors, Reverse, Hot Pink, Emerald City, and Turquoise. You can highlight any option and click the Preview button to see how the color scheme looks on your screen. Once you've made your selection, you click OK to return to the Shell screen.

## Choosing what information to display

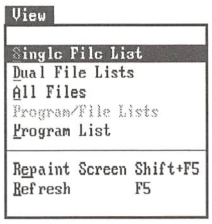
Now that you know how to make cosmetic changes to the Shell screen, let's look at how you can manipulate the display to show only the information you want to see. We'll start by examining the information the default screen in Figure A displays.

As you can see, this screen consists of a drive area, a directory tree area, a file list area, and a program list area. Although you'll sometimes find it useful to display all this information at once, you won't always need to. At times, you might want to see only the files and directories. Sometimes, you might want information about certain files. Or you might even want to see only the program list. When Shell's default screen display doesn't suit your needs, you can pull down the View menu and select a more appropriate option. As Figure E shows, the top section of the View menu offers a number of choices.

### Single File List

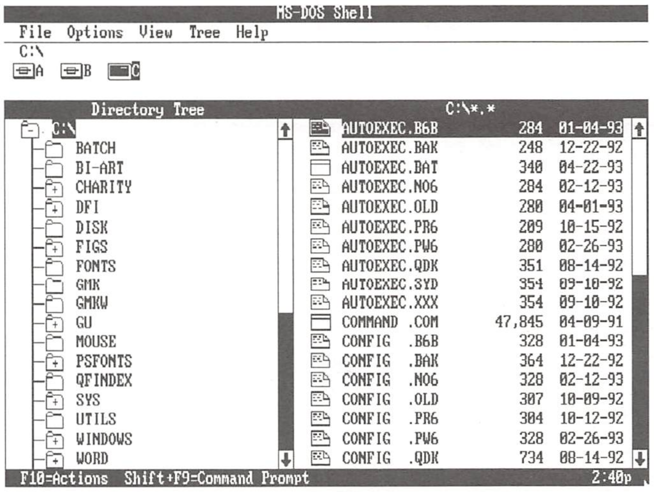
When you want to see a single list of directories and files, select the Single File List option from the View menu. Doing so hides the program list area, doubling the space for the directory tree and file list areas, as Figure F shows. Notice that the drive area remains the same.

**Figure E**



*The View menu lets you display only the parts of the screen that you want to see.*

**Figure F**



```
                              MS-DOS Shell
 File  Options  View  Tree  Help
 C:\
 A   B   C

        Directory Tree                          C:\*.*
   C:\                       AUTOEXEC.B6B       284  01-04-93
    BATCH                    AUTOEXEC.BAK       248  12-22-92
    BI-ART                   AUTOEXEC.BAT       340  04-22-93
    CHARITY                  AUTOEXEC.N06       284  02-12-93
    DFI                      AUTOEXEC.OLD       280  04-01-93
    DISK                     AUTOEXEC.PR6       209  10-15-92
    FIGS                     AUTOEXEC.PW6       280  02-26-93
    FONTS                    AUTOEXEC.QDK       351  08-14-92
    GMK                      AUTOEXEC.SYD       354  09-10-92
    GMKW                     AUTOEXEC.XXX       354  09-10-92
    GU                       COMMAND  .COM   47,845  04-09-91
    MOUSE                    CONFIG   .B6B      328  01-04-93
    PSFONTS                  CONFIG   .BAK      364  12-22-92
    QFINDEX                  CONFIG   .N06      328  02-12-93
    SYS                      CONFIG   .OLD      307  10-09-92
    UTILS                    CONFIG   .PR6      304  10-12-92
    WINDOWS                  CONFIG   .PW6      328  02-26-93
    WORD                     CONFIG   .QDK      734  08-14-92
 F10=Actions  Shift+F9=Command Prompt                            2:40p
```

*The Single File List option displays more directories and files on your screen.*

### Dual File Lists

Sometimes, you might want to display two lists of directories and files at once. For instance, suppose you're looking for a file on a floppy disk but you aren't sure which disk it's on. If you have two floppy drives, you can search two disks at a time by choosing Dual File Lists from the View menu. Figure G on page 10 illustrates the result.

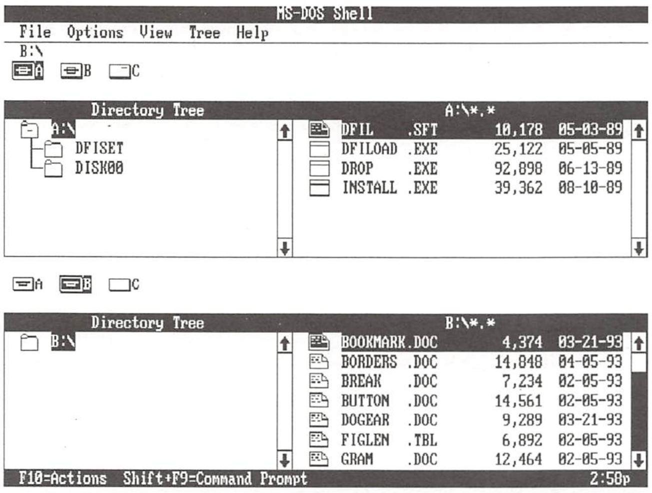
### All Files

When you want to scan through some information about a number of files on a disk, you can select the All Files option to display this information. On the right side of the screen, as Figure H (also on page 10) shows, the All Files option displays an alphanumeric list of all the files on the selected disk. The left side lists information about the currently selected file. To select the file you want to list information for, you click the filename or press ↑ or ↓ to highlight it.

### Program/File Lists

The Program/File Lists option is Shell's default—the view shown in Figure A. Keep this option selected when you want to view all the areas of the DOS Shell screen.

## Figure G

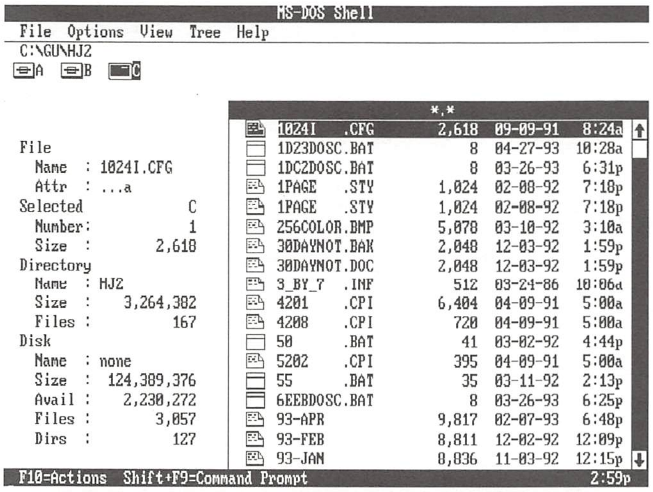


```
                                MS-DOS Shell
 File  Options  View  Tree  Help
 B:\
 A  B  C
          Directory Tree                          A:\*.*
 A:\                                DFIL    .SFT    10,178  05-03-89
  DFISET                            DFILOAD .EXE    25,122  05-05-89
  DISK00                            DROP    .EXE    92,898  06-13-89
                                    INSTALL .EXE    39,362  08-10-89

 A  B  C
          Directory Tree                          B:\*.*
 B:\                                BOOKMARK.DOC     4,374  03-21-93
                                    BORDERS .DOC    14,848  04-05-93
                                    BREAK   .DOC     7,234  02-05-93
                                    BUTTON  .DOC    14,561  02-05-93
                                    DOGEAR  .DOC     9,289  03-21-93
                                    FIGLEN  .TBL     6,892  02-05-93
                                    GRAM    .DOC    12,464  02-05-93
 F10=Actions  Shift+F9=Command Prompt                              2:58p
```

*Viewing two lists of directories and files can speed up file searches.*

## Figure H



```
                                MS-DOS Shell
 File  Options  View  Tree  Help
 C:\GU\HJ2
 A  B  C
                                                *.*
                              1024I   .CFG    2,618  09-09-91   8:24a
 File                         1D23DOSC.BAT        8  04-27-93  10:28a
   Name  : 1024I.CFG          1DC2DOSC.BAT        8  03-26-93   6:31p
   Attr  : ...a               1PAGE   .STY    1,024  02-08-92   7:18p
 Selected           C         1PAGE   .STY    1,024  02-08-92   7:18p
   Number:          1         256COLOR.BMP    5,078  03-10-92   3:10a
   Size  :      2,618         30DAYNOT.BAK    2,048  12-03-92   1:59p
 Directory                    30DAYNOT.DOC    2,048  12-03-92   1:59p
   Name  : HJ2                3_BY_7  .INF      512  03-24-86  10:06a
   Size  :    3,264,382       4201    .CPI    6,404  04-09-91   5:00a
   Files :          167       4208    .CPI      720  04-09-91   5:00a
 Disk                         50      .BAT       41  03-02-92   4:44p
   Name  : none               5202    .CPI      395  04-09-91   5:00a
   Size  :  124,389,376       55      .BAT       35  03-11-92   2:13p
   Avail :    2,230,272       6EEBDOSC.BAT        8  03-26-93   6:25p
   Files :        3,057       93-APR          9,817  02-07-93   6:48p
   Dirs  :          127       93-FEB          8,811  12-02-92  12:09p
                              93-JAN          8,836  11-03-92  12:15p
 F10=Actions  Shift+F9=Command Prompt                              2:59p
```

*You can get information on specific files by selecting the All Files option from the View menu.*

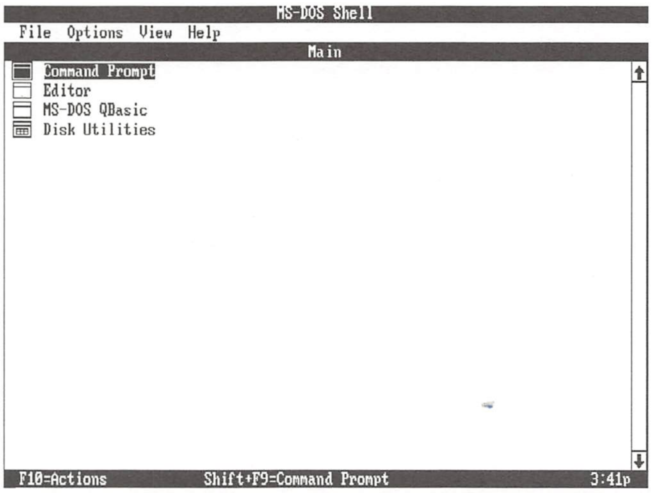
### Program List

When you aren't particularly interested in lists of directories or files but you want to run programs from the Shell, you'll want to choose the Program List option. This option maximizes the program list area, as shown in Figure I. A maximized program list area is especially useful when you're working with several program groups in the Shell. (See "Organizing Work Groups in the DOS Shell" in last month's *Inside DOS* for more on setting up program groups.)

### How task swapping affects the screen display

To have several programs open at the same time, you need to select the Enable Task Swapper option from the Options menu. Selecting this option adds the Active Task List window to your screen, as shown in Figure J. Once you've enabled the task swapper, you hide or display the Active Task List window by using the same options on the View menu that hide or display the program list area.

## Figure I

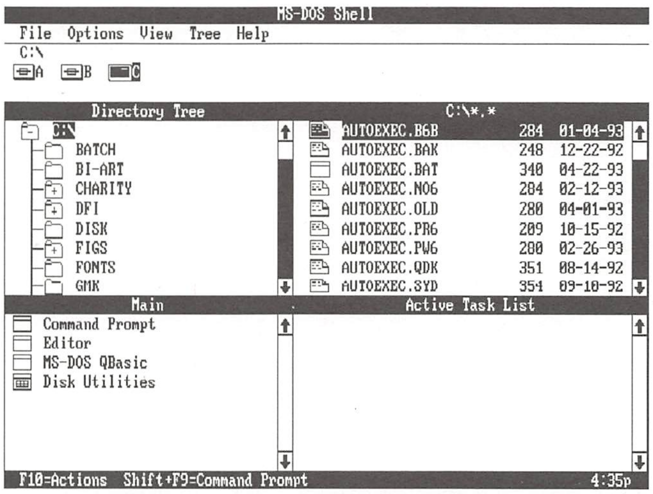


```
                                MS-DOS Shell
 File  Options  View  Help
                                    Main
 Command Prompt
 Editor
 MS-DOS QBasic
 Disk Utilities

 F10=Actions          Shift+F9=Command Prompt                      3:41p
```

*When you're working primarily with program groups, the Program List option hides everything but the program list area.*

## Figure J



```
                                MS-DOS Shell
 File  Options  View  Tree  Help
 C:\
 A  B  C
          Directory Tree                          C:\*.*
 C:\                                AUTOEXEC.B6B       284  01-04-93
  BATCH                             AUTOEXEC.BAK       248  12-22-92
  BI-ART                            AUTOEXEC.BAT       340  04-22-93
  CHARITY                           AUTOEXEC.N06       284  02-12-93
  DFI                               AUTOEXEC.OLD       280  04-01-93
  DISK                              AUTOEXEC.PR6       209  10-15-92
  FIGS                              AUTOEXEC.PW6       280  02-26-93
  FONTS                             AUTOEXEC.QDK       351  08-14-92
  GMK                               AUTOEXEC.SYD       354  09-10-92
               Main                          Active Task List
 Command Prompt
 Editor
 MS-DOS QBasic
 Disk Utilities

 F10=Actions  Shift+F9=Command Prompt                              4:35p
```

*Enabling the task swapper opens a new window to display a list of your active programs.*

### Conclusion

The Shell's graphical user interface makes it possible to customize the screen in ways that can improve your comfort and productivity in DOS. We described choosing a display mode, selecting a color scheme, and choosing the information you want to display—three general techniques for customizing the Shell screen. In future articles, we'll show you how to manipulate the directory tree and file list areas to help you get even more out of using the DOS Shell.

# Protecting your Windows swap file from DOS 6.0's DoubleSpace

*Mike O'Mara, editor-in-chief of* The Windows Report, *co-authored this article.*

Because a number of our readers have upgraded to DOS 6, we've received several requests for tips on troubleshooting problems that can arise from the upgrade. In particular, some DOS users who also use Windows have experienced a problem linked to the DoubleSpace utility, which allows you to fit more data on your hard disk. In this article, we'll outline the problem and show you ways to get around it. Since the problem and the solution center around the Windows swap file, let's begin with some background on this special file.

## The permanent swap file problem

A swap file is a hidden file that reserves space on your hard disk for Windows to use for swapping. When you run Windows in 386 enhanced mode and start running low on memory, Windows swaps information from memory to the swap file to compensate. The swap file can be a permanent file that stays on your hard disk even when Windows isn't running. Or it can be a temporary file that Windows creates in the absence of a permanent swap file and deletes when you exit Windows.

The most common problem with DOS 6.0 and Windows occurs when you add DoubleSpace disk compression to a system with an existing Windows swap file. Although Windows can use a temporary swap file located on the compressed drive, Windows can't handle a *permanent* swap file on the compressed drive.

If you attempt to create or use a permanent swap file on a compressed drive, Windows won't work. When you restart Windows to use the new swap file, Windows will try to access the file without going through DoubleSpace. The data in the DoubleSpace compressed volume file appears garbled, and you'll receive an error message similar to

```
Corrupt Swap File Warning
The permanent swap file is corrupt
```

However, you may not have a problem even if you have a permanent Windows swap file and you install DoubleSpace. When you install DoubleSpace, the installation routine attempts to detect a Windows swap file, allow room for the file on the uncompressed portion of the host volume, and make the necessary changes to your SYSTEM.INI file. The SYSTEM.INI changes are necessary so Windows can find the swap file. However, some instances (such as multiple copies of Windows or certain Windows files) can prevent the DoubleSpace installation from successfully detecting the Windows swap file. Since recovery can be difficult, avoiding the problem is a better tactic.

## The safe road

To ensure the smoothest upgrade, remove your permanent Windows swap file before installing DoubleSpace. To remove the swap file in Windows 3.1, you go through the Control Panel. To remove the swap file in Windows 3.0, you use the Swapfile application.

### Removing a permanent swap file in Windows 3.1

If you use Windows 3.1, double-click the Control Panel icon in the Main group. When the Control Panel opens, double-click the 386 Enhanced icon. Doing so opens the 386 Enhanced dialog box. Click the Virtual Memory... button to open the Virtual Memory dialog box. Then, click the Change>> button to expand the dialog box and display the New Settings options. From the Type drop-down list box, select None and then click OK. When Windows prompts you to save the changes you made to Virtual Memory, click Yes. You'll see a reminder that you must restart Windows in order for the changes to take effect. You'll also see options to restart Windows or continue. Click the Continue button. Then, choose Exit... from the File menu to quit Windows.

### Removing a permanent swap file in Windows 3.0

If you use Windows 3.0, first make sure Windows isn't running. Then, at the DOS prompt, start Windows in real mode by typing *win /r*. Once Windows is running, make sure you've closed all applications except the Program Manager. Next, issue the Run... command from the File menu. In the Command Line text box, type *swapfile*; then click OK. When the Swapfile dialog box appears, select the Delete The Current Swap File option and click OK. Swapfile will delete the permanent swap file and display a confirmation message. Then, exit Windows.

### Installing DoubleSpace

Once you've removed the Windows swap file, you can install DoubleSpace. When you do, you'll want to reserve enough space on the uncompressed volume to rebuild your permanent Windows swap file later. By default, DoubleSpace takes over all but 2 Mb of your hard drive, so you must adjust the size of the compressed volume with the Change Size command to increase the amount of uncompressed space DoubleSpace leaves.

Keep in mind that Windows doesn't want to create a swap file larger than half the available space on a drive. Therefore, if you want to create a 4-Mb swap file, you'll need to reserve at least 8-10 Mb of uncompressed space. Later, you can return to DoubleSpace to adjust the size of your compressed volume and recover the excess uncompressed space.

After removing your permanent swap file and then installing DOS 6.0 and DoubleSpace, you're ready to re-create the swap file. One note of caution: Since DoubleSpace didn't exist when Windows 3.0 or 3.1 was released, Windows can't detect DoubleSpace. As a result, Windows will offer to create a permanent swap file on a drive that's really a compressed volume created using DoubleSpace. Make sure you don't select the compressed volume as the location for your swap file.

### Recreating your permanent swap file in Windows 3.1

To recreate your permanent swap file in Windows 3.1, start Windows, open the Control Panel, and double-click the 386 Enhanced icon. In the 386 Enhanced dialog box, click the Virtual Memory... button. Then, in the Virtual Memory dialog box, click the Change>> button. Next, select the uncompressed host drive from the Drive dropdown list. As you select the drive, look at the volume labels instead of the drive letters. DoubleSpace reassigns drive letters so that C: is usually the compressed volume and H: is usually the uncompressed host drive.

Now, select Permanent from the Type dropdown list box to create a permanent swap file. Then, adjust the other settings, if necessary, so they're appropriate for your system. Finally, click OK. When Windows prompts you to save the changes, click Yes. Then, click the Restart Windows button to restart Windows.

### Recreating your permanent swap file in Windows 3.0

To set up your permanent swap file in Windows 3.0, start Windows in real mode (by typing *win /r*). Again, make sure you've closed all applications except the Program Manager. Then, run the Swapfile application by issuing the Run... command from the File menu, typing *swapfile*, and clicking OK. When the Swapfile dialog box appears, it will display information about the first drive it finds that has sufficient unfragmented space for a swap file. If the drive it finds is the compressed volume (usually C:), click the Next Drive button until information for the uncompressed host drive (usually H:) appears. Once it does, make any necessary changes to the recommended swap file size and then click the Create button. Finally, exit Windows and restart it in 386 enhanced mode by typing *win* at the DOS prompt.

## Conclusion

Since even the smallest glitch in something as fundamental as your computer's operating system can cause major problems, it's prudent to approach the upgrade to DOS 6.0 with a certain amount of caution. If you operate Windows with a permanent swap file, you might experience problems with DOS 6.0's DoubleSpace feature. To ensure the smoothest transition, remove the permanent Windows swap file before installing DoubleSpace and then recreate it on the uncompressed host drive. By doing so, you'll ensure that DoubleSpace and your Windows swap file can peacefully coexist without any interference.

**LETTERS**

VERSION 5.0 & 6.0

## Adding a directory to your path

I often see the DOS PATH command followed by *%PATH%*. What is the purpose of using *%PATH%* with this command?

*Charles Seely*
*Hammond, Indiana*

The PATH command stores the path you choose in an environment variable called *%PATH%*. You can take advantage of this environment variable when you want to add a directory to your path temporarily. For example, suppose you have a GAMES.BAT file that runs a game program that you stored in the C:\GAMES directory. You might not run this game program every day, so you won't need to add its directory to the path you set in your AUTOEXEC.BAT file. To add the program's directory to your path only when you run the GAMES.BAT file, you can place the line

```
path c:\games;%path%
```

in GAMES.BAT. Once you add this line and run the GAMES.BAT file, the C:\GAMES directory will remain in your path until you reset the path or reboot your computer.